Net2Vis – A Visual Grammar for Automatically Generating Publication-Tailored CNN Architecture Visualizations

Alex Bäuerle, Christian van Onzenoodt, and Timo Ropinski

Abstract—To convey neural network architectures in publications, appropriate visualizations are of great importance. While most current deep learning papers contain such visualizations, these are usually handcrafted just before publication, which results in a lack of a common visual grammar, significant time investment, errors, and ambiguities. Current automatic network visualization tools focus on debugging the network itself and are not ideal for generating publication visualizations. Therefore, we present an approach to automate this process by translating network architectures specified in Keras into visualizations that can directly be embedded into any publication. To do so, we propose a visual grammar for convolutional neural networks (CNNs), which has been derived from an analysis of such figures extracted from all ICCV and CVPR papers published between 2013 and 2019. The proposed grammar incorporates visual encoding, network layout, layer aggregation, and legend generation. We have further realized our approach in an online system available to the community, which we have evaluated through expert feedback, and a quantitative study. It not only reduces the time needed to generate network visualizations for publications, but also enables a unified and unambiguous visualization design.

Index Terms—Neural networks, architecture visualization, graph layouting

1 Introduction

Papers utilizing CNNs are published on a daily basis. An essential aspect of all these publications is to communicate the used or developed network architecture. Accordingly, a rising number of architecture visualizations can be observed from year to year (see Fig. 2). Authors, who often may lack visualization expertise, mostly use handcrafted, non-standardized visualizations. As a consequence, generating visualizations takes significant time, and authors often employ suboptimal visual encodings that are sometimes even inaccurate or erroneous.

We argue, as backed by our expert questionnaire (see Section 6), that the time invested in suboptimal visualizations would be better used to improve training results. Nevertheless, such abstract visualizations are generally considered to be of great importance. Therefore, automated approaches that obey to a common visual grammar are required. We argue that, ideally, such a visual grammar should be informed by three factors: current practice, expert demands, and visualization expertise. Accordingly, we have analyzed properties of existing architecture visualizations, which we scraped from all ICCV and CVPR papers published between 2013 and 2019 – which led to a pool of 751 such visualizations. ICCV and CVPR are prime conferences on machine learning for vision-related tasks and, thus, reflect the great need for such automated visualization approaches. Additionally, we contacted authors of highly cited papers encompassing architecture visualizations, in order to assess their demands. Last but not least, we brought in established rules from the data visualization literature to inform our visual grammar. Based on this, we propose the first method to automatically generate abstract, publication-tailored visualizations of complex, modern CNN architectures, obeying a unified visual grammar, which we refer to as *Net2Vis*. To this end, we make the following three main contributions:

1) We propose a set of requirements for effectively communicating neural network architectures, based on expert feedback and the analysis of existing visualizations.
2) Based on these requirements, we propose a new visual grammar for CNN architecture visualizations, which we make available via an online platform that transforms Keras code into visualizations tailored to the use in publications.
3) We release a data set of 751 neural network architecture visualizations, which we have extracted from all papers published at ICCV and CVPR between 2013 and 2019.

Fig. 1 shows an example visualization of a U-Net variant generated using our approach. To evaluate our approach, we conducted both a quantitative user study and a qualitative usability evaluation. The obtained results indicate that our techniques are beneficial for creating and reading CNN

• *The authors are with the Visual Computing Group at Ulm University, 89081 Ulm, Germany. E-mail: {alex.baeuerle, christian.van-onzenoodt, timo.ropinski}@uni-ulm.de.*

Manuscript received 28 May 2020; revised 26 Jan. 2021; accepted 3 Feb. 2021. Date of publication 8 Feb. 2021; date of current version 29 Apr. 2021. (Corresponding author: Alex Bäuerle.) Recommended for acceptance by S. Liu. Digital Object Identifier no. 10.1109/TVCG.2021.3057483

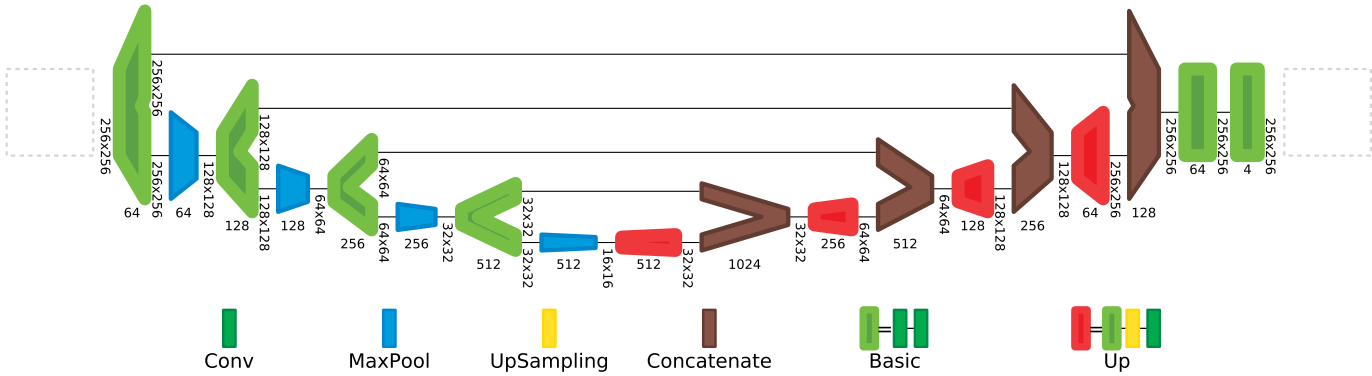

Fig. 1. Visualization of a U-Net variant. It was automatically generated using our approach based on the Keras code describing the architecture. Data flows from left to right. Glyphs represent layers or aggregates, while lines represent connections. Glyph widths communicate feature size, while heights communicate the spatial resolution. Both values are also given through labels, while dashed boxes on the left and right serve as placeholders to provide input and output samples. The legend communicates layer types and the composition of aggregates.

architecture visualizations, which is important for broad acceptance and unambiguous CNN architecture communication. Our techniques can be used in the form of an online platform at: https://viscom.net2vis.uni-ulm.de.

2 Related Work

Handcrafted visualizations are part of many research papers that use neural networks in their publications [1], [2], [3], [4]. However, they differ greatly in their visual appearance, which complicates transferring knowledge between them, e.g., [5], [6], [7]. In addition, they sometimes contain errors, as can be observed in work done by Henzler *et al.* [8], where visual glyph encoding and glyph labeling diverge. Thus, automatically visualizing network architectures to convey their underlying ideas is an extensive field of research. In the following, we divide related research into approaches for debugging and investigating network architectures, and approaches targeted towards communicating these.

Debugging Approaches. Demonstrating the importance of visualization for the field, most deep learning frameworks, such as Tensorflow [9] with TensorBoard [10], Caffe [11] with Netscope [12], and also Keras [13], directly provide visualization toolkits.

All of these are clearly designed for online use. They are based on vertical layouts for detailed visualizations including all layers and parameters and provide some information only on interaction. Their consumption of visualization space and required user interaction are perfect for debugging the network architecture, but it renders them inapplicable for use in publications.

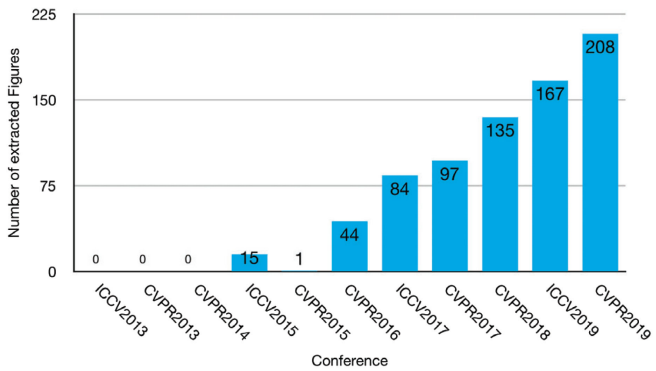

Fig. 2. Number of CNN architecture figures we extracted from all ICCV and CVPR papers between 2013 and 2019. We searched for pages of papers containing figures and the words *figure* and *architecture* in the same line to extract these. Then we manually filtered them to obtain only neural network architecture visualizations.

Network visualization tools similar but unrelated to these frameworks such as ANNvisualizer [14] and Netron [15] suffer comparable shortcomings. Their glyphs do not convey any information apart from layer type, whereby additional information is displayed by overlaying textual annotations on top of the used glyphs. Additionally, their vertical layout, along with spacing between layers makes even small networks appear relatively large.

Another interesting visualization approach along this line was presented by Wang *et al.* [16]. Here, the focus is on comparing different neural network architectures. The approach can be used to identify differences in neural architecture design, compare the number of parameters, and draw conclusions for one's own architecture choice. However, while this approach allows for in-depth comparisons through interactive visualizations, it is not designed to convey network architecture details in a compressed, static way.

Communication Approaches. Some visualizations convey neural network architectures to explain their functionality to novices [17], [18], [19], [20], [21], or are targeted towards analyzing what a network has learned [22], [23], [24], [25]. These visualizations clearly fulfill their purpose to support education or interpretability, but are not designed for use in publications. They all display basic network architectures limited to a specific use case and are not generalizable to more complex architectures.

One visualization technique that is specifically targeted towards use in scientific papers is Drawconvnet [26]. Convnet-drawer [27], which builds on the aforementioned, provides such visualizations, and even allows visualization generation from source code. Similarly, NN-SVG also claims to create publication-ready network visualizations [28]. While these techniques can be used for small and simple networks, they all face major problems. First, they do not scale to modern, large network architectures since no aggregation technique is implemented. Second, they visualize layer connections simply by placing the layers from left to right, which means that parallel network parts cannot be

represented. Additionally, in Drawconvnet and NN-SVG, users have to invest the time to rebuild their network architecture to obtain visualizations.

The surveys by Hohman *et al.* [29] and Yuan *et al.* [30] discuss many of these graph visualization techniques. One important downside of all currently available approaches is that they struggle to visualize large networks in a compact way. Thus, it remains an open challenge to generate publication-tailored visualizations, despite the existence of the visualization systems described above. Current state-of-the-art visualizations [10], [12], [15] allow to inspect operations in great detail. However, these visualizations lack abstractions to make the general network structure comprehensible at a glance. For a demonstration of this problem, see our supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2021.3057483, which contains a comparison between Netron, Netscope, TensorBoard, and our approach. Other visualization techniques that aim at providing publication-ready visualizations cannot handle modern network architectures [26], [27], [28] and lack important features requested by experts. Thus, in research papers, these complex networks are usually simplified and drawn manually [4], [7], [8].

Besides the extra time effort related to this manual drawing process, the field lacks guidelines to create such visualizations, as is observable in our review of papers from CVPR and ICCV. Some properties in existing visualizations are ambiguously interpretable, e.g., where downsampling happens, and, for the lack of a common visual grammar, knowledge can hardly be transferred between different publications. Therefore, we propose a novel visualization technique for abstract architecture visualizations that are optimized for use in scientific publications, where display-space is limited and interaction is impossible.

3 Design Requirements Abstraction

We argue that for abstract CNN visualizations, both model properties and layer properties need to be visualized. Model properties are important for the layout and arrangement of the network architecture, while layer properties visualize parameters of individual layers, or groups of layers. In the following sections, we describe how we used the analysis of current practice and insights from visualization research to inform our design proposition for CNN architecture visualizations.

Data Collection. To support this analysis, we interviewed multiple ML practitioners and reviewed 751 figures of neural network architectures extracted from papers of all CVPR and ICCV conferences between 2013 and 2019. This data was gathered by crawling http://openaccess.thecvf.com/ for all 7988 main conference papers using scrapy [31]. We then filtered for neural network architecture visualizations and extracted all 1168 pages that contain a figure and had a line of text containing both the words *figure* and *architecture* using PyPDF2 [32]. We used pdffigures2 [33] to then extract the figures from these pages. This yielded 1027 images which we manually coded with respect to their visualization design choices. Since some of the used tools did not work perfectly, we had to delete images that did not contain, or contained corrupted versions of architecture visualizations, leaving a total of 751 figures. To our knowledge, this is the first data set of such visualizations. The visualization properties we analyzed are italicized in the following sections and listed in the data set we release alongside this paper. The following observations are based on both, expert feedback and this figure analysis. When based on figure analysis, we indicate how many of the analyzed papers used a certain encoding as a percentage value.

3.1 Model Properties

In the following, we discuss which model properties of a neural network need to be communicated to quickly assess relevant architectural decisions within a neural network, as derived from the collected data.

Layout. Maybe the most important factor when visualizing neural network architectures is the interconnection of layers as it communicates the order in which the computation graph is executed. At the same time, there is also limited space for publication figures, which is to be taken into account when designing such visualizations. Thus, one has to find a layout that clearly communicates the order of computations, while also not wasting too much space for a single publication figure.

Connections. A consistent layout helps to resolve the order in which the network graph is traversed, but there is still important information missing, namely, connectivity. Without connectivity information, it is not clear which layer in a network routes data to which following layers. Especially if network graphs contain parallel execution steps, simply laying out the network layers in a consistent manner will not always resolve which layers actually interact with each other. Thus, communicating which layers are connected in the computation graph is essential.

Aggregations. When thinking about visualizing modern network architectures in publications, space and complexity is a major point of concern. It is a well-known fact in the visualization literature that hierarchical *aggregation* can help to simplify visualization designs [34]. Following this, as networks get more complex, authors manually aggregate layers to make their architectures fit on one page (63.4 percent). Here, *legends* sometimes indicate which layers are aggregated (15.2 percent). Both of these numbers were rising over the years, as shown in our supplementary material, available online. We, thus, argue that communicating modern network architectures requires a way of aggregating layer glyphs to create overview visualizations.

Omission. Another way of simplifying displayed network architectures is to just omit layers that are not important to convey the general idea of the network architecture. This is supported by our expert feedback, which indicates that simplification is a major feature for visualizations of neural network architectures.

Input and Output Samples. Several of the network visualizations incorporate *input or output examples* (73.1 percent). However, samples are mainly useful for image or shape related tasks and do not provide additional information concerning the network architecture. We thus argue that such samples can help for some application areas, while they should not be presented for others.

3.2 Layer Properties

Layers are the building blocks of the computation graph that defines any network architecture. Thus, visualizing properties that parametrize these layers is important to convey the structure and architectural decisions of said network. In the following, we discuss important layer properties and explain which of those should be visualized for obtaining a general overview of a network architecture.

Layer Type. We consider the *layer type* to be the most important layer-specific variable. Together with the model properties, it already helps greatly in determining the functionality of a neural network. It is thus important to encode layer types as a visually prominent variable.

Spatial Resolution. Next, to determine the functionality of a neural network architecture, it is important to be able to follow the transformation of data into or out of the latent space, which is often referred to as the change of spatial resolution. Thus, the dimensionality of data is another variable to be considered when visualizing neural network architectures. While only slightly more than half of the surveyed visualizations encode the *spatial resolution* (53.7 percent), expert feedback, and the fact that this variable can be encoded within the layer glyphs, suggests that visualizing it brings great benefit at no cost. We thus advocate for always providing information about spatial resolution.

Feature Channels. All previously mentioned attributes apply to almost all types of neural networks. Feature channels, on the other hand, are mainly important for CNNs. Thus, most visualizations do not at all encode the number of feature channels (56.7 percent). Since feature channels match the variable type of the spatial resolution, and since they are tightly coupled across the network, they are often viewed in combination. Thus, they support the assessment of data transformation, and our evaluations surfaced that their display is important for architecture visualizations of CNNs.

3.3 Properties Not to Be Visualized

We propose a set of properties to be visualized for providing an overview of a network architecture. However, there are other properties we explicitly advise to omit in non-interactive visualizations of neural network architectures.

Kernel Size. Kernel sizes can be found in many visualizations as textual descriptions of layers (24.4 percent) or as encodings in the layer glyphs (5.0 percent), but are not encoded in most of the visualizations (73.5 percent). When analyzing why kernel sizes are not displayed in most visualizations, two factors were apparent. First, kernel sizes often are consistent across multiple layers leading to repeated information. This is in contrast with the request for reduced complexity by domain experts (see Section 6). However, the biggest problem with them is that they are in stark contrast with layer aggregations, which are important to reduce the complexity of network visualizations. For aggregations, there is no such thing as one kernel size as it may differ for layers contained in them. Additionally, as aggregations are reused, kernel sizes may be different again. We aim at reducing repetition and embracing aggregation. Thus, we propose not to display kernel sizes in overview visualizations for neural networks. This is in line with the expert feedback of domain experts, e.g. *I'm very interested in great visualization tools that emphasize function and architecture over the kind of "obtuse prettiness" [...].*

Additional Layer Properties. Neural network layers contain many more features such as weights, strides, padding, and others. Yet, most of them only exist for certain layer types and are thus rarely communicated. Some printed visualizations include features such as activation maps, e.g., [5], or receptive fields, e.g., [35], but they are only provided for specialized use cases. In contrast, we argue that for obtaining a general network architecture overview, such features are not necessary. This is supported by our expert feedback and in line with our goal of providing an abstract overview of the network architecture, where detailed information can be obtained by reading the publication it is contained in.

Dimensionality. Three-*dimensional* visualizations are helpful if the reader's task includes shape understanding, but less so for any relational task [36], [37]. Thus, the relation of layers and their spatial position would suffer from being visualized in 3D, while the benefit of using three-dimensional layer glyphs would only be to resemble the shape of data in case it is three-dimensional. Another important argument against three-dimensional visualizations in publications is that the viewpoint cannot be changed. Contrary, exploration is essential for utilizing the benefits of three-dimensional visualizations [36], [38]. About half of the analyzed network visualizations display layers in 3D (50.2 percent), however, this added dimension does not convey additional information for most of them. Since the reduction of spatial resolution is almost always applied to all spatial dimensions equally, the third dimension is not needed for the visualization of such changes. For those reasons, we advise not to visualize layer glyphs in 3D.

3.4 Summary

Based on these requirements, we propose to use the following properties in any visualization that aims at providing a general overview of a CNN architecture in non-interactive visualization environments:

Model Properties

1) Layout
2) Connections
3) Aggregations
4) Omission
5) Input and Output Samples (for some tasks)

Layer Properties

1) Layer Type
2) Spatial Resolution
3) Feature Channels (for convolutional layers)

We, thus, elaborate on *what* to visualize in this section, defining the dimensions of our proposed design space. This assessment is based on our analysis of figures extracted from the top conferences in the field, as well as expert feedback both before and during the development of our approach. It is strictly tailored towards conveying the overall idea of a network architecture, and does not cover cases in which specific features of a network are to be communicated. Therefore, we also provide guidelines for which features not to visually encode for abstract architecture

visualizations, thus preserving simplicity and preventing the need for interaction. By always visualizing the aforementioned parameters in the same way, users can transfer knowledge between different visualizations. We thus advise not to render the mapping of variables customizable. Instead, we propose a unified visualization design for these parameters. In the following sections, we map these visualization properties (i.e., *what* to visualize) to specific visual encodings (i.e., *how* to visualize them), describing the proposed design space inspired by similar design space descriptions [39], [40].

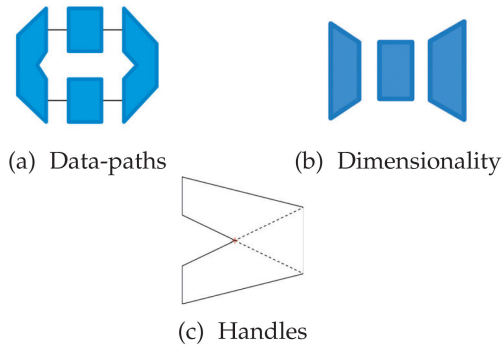

Fig. 3. (a): Two simultaneous data-paths. The layer displayed on the left has two outgoing connections, while the layer displayed on the right fuses these paths back together. (b): Left: layer that reduces the spatial resolution; Center: same spatial resolution for input and output; Right: a layer that increases the spatial resolution (c): Multi-handled glyphs are always connecting corners on the input side with corners on the output side, here depicted by a dotted line.

4 Visualization of Model Properties

Based on our assessment of important visualization aspects for communicating CNN architectures as described in the previous section, we now explain *how* we propose to visually encode the model properties. For these global properties, the design space consists of the aforementioned dimensions, i.e., Layout, Connections, Aggregations, Omission, and Data Samples, which we describe in the following subsections.

4.1 Layout

The representation of this design space dimension mainly concerns the spatial arrangement of layers which can be vertical or horizontal, and in any direction. Most investigated neural network visualizations layout their layers from left to right (81.6 percent). Our figure analysis as well as the collected expert feedback indicate that for publications, this layout is preferable over vertical layouts, which are often used in online tools. It does not only preserve the reading direction of western cultures, but visualizations also nicely fit across the width of a page. Furthermore, perceptual rankings indicate that position best encodes ordered data [41], [42], [43], such as the order of network layers. Follow these insights, and the fact that space taken up by publication figures is important, we propose to employ a narrow horizontal layout, in which parallel execution steps of the network are stacked vertically on the same horizontal position.

To layout CNN graphs, we propose to use the network simplex algorithm [44] which is explicitly targeted towards drawing directed rank-based graphs. The rank-based nature of this algorithm perfectly fits our use case in which parallel layers are to be placed at the same x-coordinate, and sequential parts of the network tend to be drawn on the same vertical level. Using an algorithm that only layouts series-parallel graphs is not an option, since Keras operations are not restricted to those (e.g., $[a \rightarrow b, a \rightarrow c, b \rightarrow d, c \rightarrow d, d \rightarrow e, b \rightarrow e]$).

4.2 Connections

For this dimension of the design space, possible representations are an implicit connectivity without explicit visualizations and different forms of connecting lines. In existing figures, connections between layers are visualized either using lines (73.4 percent) or by simply placing layers next to each other. Some visualizations additionally add arrowheads to clarify the *direction* of data flow (65.9 percent). Following most visualizations, we also propose to use lines as connections between layers to emphasize the graph structure of neural networks. However, we propose not to add arrowheads to these connections, as the forward data flow direction is uniformly left-to-right in our visualizations.

Many architectures contain *skip connections* visualized by lines between distant layers (55.9 percent, rising over the years, see supplementary material available online). Displaying splits in the execution graph only through lines has the negative implication that size-related attention bias is induced [45], [46]. Thus, we propose a glyph design that prevents such issues. Whenever a layer has multiple outgoing or incoming connections, we modify the glyph that represents it as shown in Fig. 3a. This way, there might be multiple ends on the left or the right side of the glyph each having the same visual prominence. At the same time, splits and joins of the data flow, which are important features of the architecture, are highlighted. A visual representation of such multi-handled glyphs is illustrated in Fig. 3c.

One might think that this layer shape induces problems with edge crossings. However, edge-crossings are uncommon to neural network architectures. We have only found planar network graphs which consequently can be laid out to avoid crossing edges.

4.3 Aggregation

The aggregation of multiple layers is another dimension in our design space. Here, representations can be no aggregation of layers, vertical stacking, or replacement with a group-representing glyph. As modern networks become increasingly complex, the aggregation of layers is inevitable. Such aggregations can contain many layers at once, and even parallel paths, which is why we do not employ a stack-based visualization. Instead, we adopt the paradigm of providing ways to aggregate multiple layers and replacing them with a new, single glyph. As this removes direct insight into the content of aggregations, we resolve them in a legend below the network graph. Furthermore, domain experts and users frequently requested a visual separation of aggregations. Thus, in our approach, their border is drawn thicker, and their color scheme is inverted (lighter border than center).

Aggregation Constraints. As aggregations substitute all occurrences of selected layer sequences throughout the

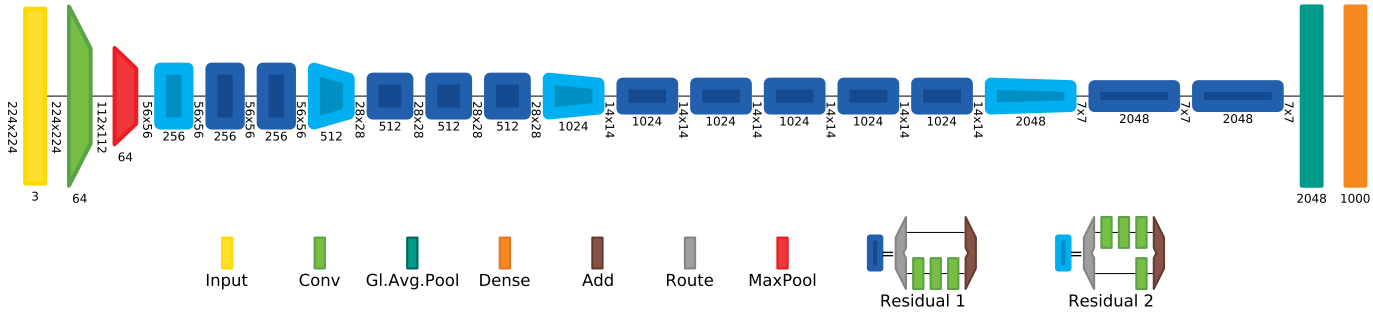

Fig. 4. With ResNet50, the removal of activation and batch-normalization layers from the visualization, which do not add information about the network structure, along with the repetition of residual blocks in ResNet allows us to reduce the number of glyphs that need to be drawn from 177 to just 21, even though we added routing layers to clearly define the beginning of a residual block.

graph, only parts of the network graph that can be represented by one sequential layer can be aggregated. We, thus, restrict aggregation to sequentializable segments. This way, no deformed aggregation layers can occur, where two outputs or inputs might differ in their spatial resolution. This ensures visual consistency such that layers always manipulate the data the same way for all of their connections and that all connections of each layer end on the same horizontal level in the graph.

Automatic Aggregation. To generate aggregations, the layers to be aggregated can either be selected by the user or more conveniently be selected automatically. To obtain automatic aggregations, we propose to analyze all sequential parts of the network. We then search for recurring sequences of layers. The most frequent of these sequences is then assumed to be the preferred aggregation.

Interacting With Aggregations. We argue that visualization designers should be able to remove or temporally deactivate aggregations of the network, which expands abstracted layers back to their initial layout. Deactivated aggregations are preserved in the legend for later reuse to be able to explore the visualization without losing information. To visually convey the state of aggregation, we propose to draw active ones with a dark outline and black description text, while outlines and text are drawn in light gray for inactive aggregations and layer types that are hidden by the user.

This way, the effect of different aggregation levels can be explored while preserving all aggregation information.

Split Layers. While there are dedicated layers to fuse computation paths, routing the data to multiple outputs is done implicitly. Thus, it is possible, that, e.g., an activation layer feeds data to two different paths. Visually, this is a problem as splits and merges of the computation graph are often seen as blocks and frequently get aggregated by the user. Activations are orthogonally seen as the end of a computation group rather than a start of one. For such dedicated groups, we argue that the user should be able to add special routing layers. This way, they can clearly communicate the special role of such multi-path aggregations while also assigning more importance to data path splits, e.g., as shown in Fig. 4.

4.4 Omission

In the computation graph, any function that has been added by the developer is considered a network layer. However, these graphs can be defined at different levels of detail (e.g., activation within layer or separately). Thus, in our approach, this graph can be thinned by the user to better convey the underlying architecture rather than each individual computation step by hiding individual layer types entirely. Showing a graph overview, and then allowing the user to filter it is in line with Shneiderman's mantra [47].

4.5 Input and Output Samples

As described in the property analysis of Section 3, input and output samples may, for some networks, be helpful. Directly integrating such samples would, however, require the user to provide training or testing data, and thus interfere with the automatic nature of our visualization design. We thus propose not to include them directly in any programmatic CNN architecture visualizer. Instead, we suggest to optionally provide placeholders for input and output samples which users can replace during post-processing with actual samples.

5 Visualization of Layer Properties

In the following, we describe the design space dimensions of the visual layer encoding we propose based on the design discussion presented in Section 3, in other words, *how* to visualize layer properties. Our visualization design supports the direct encoding of layer type, spatial resolution, and the number of feature channels. By reducing the visualization space to these three variables, we are able to encode all of them in simple glyphs that represent the layers of the network architecture to be visualized. For the spatial resolution and the number of feature channels, percentages only consider the 464 visualizations that contain convolutional layers, as we explicitly tailored our glyph design to work well for CNNs.

5.1 Layer Type

The design space dimension for the layer type can take the representations of textual display, color-coding, and shape-encoding. Most visualizations only use textual descriptions (52.7 percent), or text along with glyph color (14.6 percent) to convey layer types. However, it can be repetitive to encode the layer type as text below each layer. We, thus, propose to provide this textual encoding optionally (see Fig. 6) while not being displayed in the default setting. The layer type is a categorical attribute and consequently best visualized using a channel that is optimized for such data [36], [48]. In Mackinlay's ranking [41], color ranks just

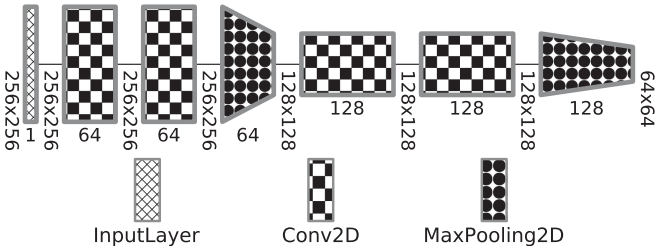

Fig. 5. Accessible encoding of the layer type to also support readers with monochromatic vision and publications without color.

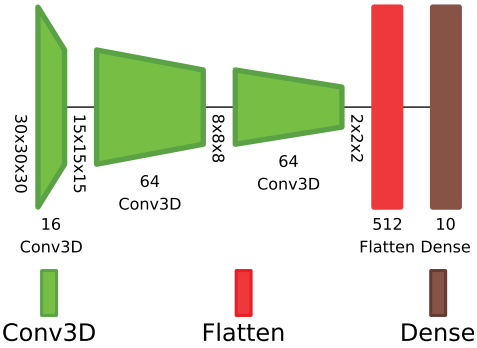

Fig. 6. *Net2Vis* can also be used to generate visualizations of multi-dimensional network architectures.

behind position, which we already employ to communicate the data flow of the network. Perception research also shows that color is a visually dominant channel [49]. Thus, we propose to use a color encoding to convey the layer types in the visualized network.

Color Assignment. In our approach, colors for new layers are automatically preset in accordance to one of two alternative approaches. The first approach is motivated by farthest point sampling. It finds unused colors in hsv color space by searching for the biggest gap between any two hue values of already used colors. This is the most functional approach, as it optimizes for color difference. Unfortunately, it might result in colors that are indistinguishable by color-blind users and unpleasant color choices. Therefore, the second option for color proposition is palette-based and serves as the default. We suggest to use two color palettes, one from materialuicolors [50] for visually pleasing color mappings, and one adapted to users with trichromatic or dichromatic vision [51]. Additionally, visualization designers should always be able to customize the color that is used to encode a layer type.

To also make our visualizations accessible to readers with monochromatic vision, and support publications without colored images, we also propose a texture-based encoding of layer types, see Fig. 5. We provide twelve distinguishable patterns that can be extended upon when needed.

Legend Generation. Since we use color-coding to differentiate between layer types, a legend that maps these color codes back to layer names is needed. This legend contains a glyph for each layer type in the network and displays the name of its layer as shown in Fig. 4. Based on expert feedback, legend items are sorted from simple to complex. This complexity is determined by analyzing the dependency-tree of aggregations, whereas nested aggregations are seen as more complex.

5.2 Spatial Resolution

The spatial resolution is a design space dimension that might be represented by glyph height, glyph width, glyph color, and textual annotations. As the spatial resolution is a quantitative and sequential variable, length, angle, slope, and area are the best remaining options for encoding it [41], [42], [52]. In some visualizations, the spatial resolution is represented by the shape of the layer glyph in combination with textual information (10.8 percent) or just textual representations (10.1 percent). However, mostly only glyph shape (32.8 percent) is used. We propose to use height in combination with text as a direct mapping of the spatial resolution, as this does not interfere with our network layout and can be encoded in the glyph design directly.

However, not all visualizations map glyph height in the same way. Sometimes the height of the downsampling layer already encodes the changed resolution, while in other visualizations, the next layer is first affected by this change. This ambiguity makes the interpretation of such visualizations hard, as one needs to determine which representation was chosen for each visualization approach. Furthermore, the transformation of the resolution is determined by multiple parameters (e.g., stride, kernel size, padding). Thus, the output resolution is a result of the inner working of a layer rather than a fixed parameter. We, therefore, propose to visualize the spatial resolution as a change within the layer. To convey the underlying transformation, we set the height on the left edge of the glyph to match the input resolution, while the height on the right edge of the glyph reflects the output resolution, resulting in trapezoid-shaped glyphs, as shown in Fig. 3b. This conveys the change of resolution as a result of the mathematical operations within the layer while at the same time removing its ambiguity. In addition, with this encoding, the relation of input and output dimension from layer to layer as well as across the whole network is clearly visible by horizontally scanning the visualization [53].

To draw these glyph shapes, in our approach visualization designers can define a minimal and maximal height for the glyphs. We then obtain the spatial resolution for the input and output tensors for each layer. The highest and lowest value of all spatial extents gets mapped to the extremes of the user-defined height values. Values between these extremes are interpolated linearly to convey the actual spatial resolution for the input and output of each layer in the network. Using these interpolations, each input and output of every layer gets a height value assigned, which maps this important quantitative variable to the vertical length of the glyph ends, as Mackinlay's ranking suggests [41], [52].

We found that it is mostly not important to convey the exact spatial resolution by means of textual descriptions, as only 20.9 percent do so. Since many modern architectures further allow inputs to be arbitrarily shaped, e.g., [3], [7], [54], the spatial dimension is not necessarily fixed at any given layer for many network architectures. We, therefore, advocate for the option to toggle labels that display the exact spatial resolution between the layers, following our visualization design, in which the resolution is fixed between layers but changes within them.

5.3 Feature Channels

Similar to the spatial resolution, feature channels may be represented by glyph height, glyph width, glyph color, and textual annotations. Textual descriptions (20.0 percent), glyph shapes (13.8 percent), or a combination of both (9.1 percent) are common for conveying the number of feature channels. The number of feature channels, just as the spatial resolution, represents the important transformation into or from latent space, tightly coupling these two variables. Thus, we propose to employ a similar visual encoding to convey them, again, mapping a perceptually preferable length parameter, in this case, glyph width, to this variable [41], [43]. Feature channels are different from the spatial resolution in that they are fixed properties of a layer and not derived from the previous dimensions. We, therefore, propose to visualize this variable as a direct property of the layer rather than a change within it. In our approach, the number of feature channels can additionally be represented as text, displayed below each layer.

While the spatial resolution and its change within a layer is represented by the heights at both ends of the layer glyph, combining this representation with the number of feature channels as encoded in the glyph width can reveal important overall information on the processed data. Together, they present the total amount of data that is processed by the layer. In our visualization design, this amount of data is implicitly encoded in the area covered by the glyph.

Accordingly, with this glyph design, variable importance is perfectly aligned with Mackinley's ranking [41], [42], [53]. At the same time, these glyph shapes fit nicely into the horizontal network layout.

Dense layers only have one intrinsically specified dimension of data. Since this is a fixed dimensionality, it is more similar to feature channels than the spatial resolution. Additionally, dense layers are one-dimensional and commonly visualized as vertical chains of neurons. Thus, for dense layers, the number of neurons is also mapped to the glyph height. Additionally, for better differentiation of these simpler layer types, we also propose to employ a simpler color-coding, using the same color for the border as well as for the body of the glyph.

For all of these visual encodings, we propose default specifications, such as minimal and maximal width and height of layer glyphs, which can be customized by the user.

6 Evaluation

To evaluate the proposed concepts, we have visualized multiple well-known architectures with our techniques, gathered expert feedback to inform and evaluate our visualization design, and conducted a quantitative user study.

6.1 Application Examples

To provide the community with means to incorporate our visual encoding, we implemented *Net2Vis* as an online application which is available at https://viscom.net2vis.uni-ulm.de. Here, users can paste Keras code to obtain ready-to-use architecture visualizations and download those as PDF figures for direct use in publications, as well as SVG images, which can be edited in hindsight. To demonstrate *Net2Vis*' capabilities, we applied it to several commonly used network architectures. Fig. 1 shows a variation of U-Net [55] which is frequently used for semantic segmentation. Fig. 4 shows a visualization of ResNet [56] where we show a reduction from 177 to just 21 glyphs through our aggregation techniques. Finally, in Fig. 6, we demonstrate that we also support multi-dimensional network architectures. Even more application examples where we show that our techniques can even visualize networks such as InceptionV3 [57] can be found in our supplementary material, available online. Here we show popular published neural network architectures [1], [6], [55], [56], [57], [58], [59], [60], [61], [62], [63], and two network visualizations we used for our own publications and presentations.

6.2 Comparison to TensorBoard

TensorBoards graph visualization is based on the computation graph as defined in the program code for the network architecture. Aggregations in the graph visualizer can, thus, only be made for those nodes that contain sub-nodes. This can provide more detail as every operation in the graph can be examined. However, TensorBoard's aggregation approach is far less flexible and not tailored towards publication figures. For publication visualizations, users typically want arbitrary aggregations of multiple layers without having to specify parent nodes for those in the code first. Our flexible graph-based aggregation methods support that exact need, in that they allow for tailoring aggregations in a way that best communicates the overall architectural ideas. This also leads to much smaller and easier to understand figures, which we will elaborate more on in the following evaluations.

6.3 Expert Interviews

After the implementation of the first version of *Net2Vis* was complete, we conducted qualitative interviews with experienced machine learning researchers.

Expert Selection. We used Net2Vis to generate replications of visualizations from published papers. These visualizations were then emailed together with our questionnaire to the authors of the respective papers (in the following referred to as experts). In total, we contacted 7 experts in this manner who had novel papers or high citation count and published at different conferences.

Three of those answered our questions, one replied to have other obligations, and three did not get back to us. Two of the papers from which the respective experts gave feedback are highly cited (i.e., Noh *et al.* [1]: > 2300 and Long *et al.* [6]: > 14000), the third is a more recent publication from 2018 [59].

Questionnaire. Our questions were designed following Munzner's nested evaluation model [64]. Thus, we assessed the need for such automatic visualizations (Q1, Q5), analyzing the threat of targeting a wrong problem. We also investigated why 3D visualizations are so common (Q3), and asked about our visualization design (Q2, Q4) to evaluate the abstraction and encoding technique, which are the second and third possible pitfalls [64]. Our implementation proves interactivity, the fourth possible visualization pitfall [64].

We intentionally asked only five questions to keep the time needed to answer our survey relatively low, and thus

maximize the chance for responses from these well-known researchers.

However, we encouraged the experts to add any comments to their replies.

Feedback. All replies were positive and emphasized the importance of such automatic visualizations. Furthermore, they gave positive feedback on our glyph and graph design. Their main concerns were scalability to different architectures since they only received visualizations for their papers. However, as can be seen in Section 6.1 as well as the supplementary material, available online, *Net2Vis* is designed to work with a wide variety and high complexity of convolutional neural networks.

Q1: How Much Time Did You Spend Creating Your Figure? All experts stated that creating figures of their architecture took too much time. While initial versions seemed to take about one hour on average, they all noted that they needed multiple iterations.

This supports our claim that this task can be greatly optimized as it takes valuable research and paper-writing time from the experts.

Q2: How Do You Understand the Mapping of Number of Feature Channels and Spatial Resolution in the Visualizations We Sent You? All three responded that they understand that and how the spatial dimension and feature channels are mapped onto the glyphs correctly. Thus, the mapping of spatial dimension and feature channels seems comprehensible even though this representation is more abstract than the ones used in their papers.

Q3: Why Did You Pick a 3D Visualization for the Layers and Which Information Did You Want to Convey? All experts said this was done since the data was three-dimensional. None of them conveyed additional information this way. One expert also admitted that 3D visualizations introduce the problem that layers cannot always be evenly spaced because of occlusion.

All three also noted that this makes the visualization more complex. Our choice of visualizing the network architecture in 2D was preferred also by these experts.

Q4: What Do You Think of Visualizing The Transformation That Happens During Pooling/Unpooling as a Transformation of the Layer Itself (Trapezoid Glyphs) Rather Than In-Between? One expert said *I found many people complained about not drawing in-between relation between pooling/unpooling*, which indicates that this implicit transformation used in the existing visualizations is confusing to the reader. Another expert mentioned that *the trapezoids seem like a nicely simple way to indicate where and how much downsampling is going on*. However, he also noted that this is dual to the way AlexNet visualizes network architectures which has been picked up by many researchers. While it is a valid concern in that readers have to differentiate between these approaches, we think that the mentioned benefits outweigh the downsides which naturally come with adopting a new visualization approach.

Q5: Would You Use Such a Tool For Your Projects, if Available? All experts agreed that they would be users of network visualization generators as proposed in this paper. One expert additionally mentioned that he would still want to have the possibility to modify the visualization to his will which he did not know is possible in *Net2Vis*.

Other remarkable comments that clearly show the need for such automatic visualizations were: *I have been ranting about this for a while and have been waiting for somebody to ask*, and *I've been hoping someone would make better automatic visualization toolkits for a while*.

Based on the comments and free-form texts within the expert feedback, we further refined our visualization design, e.g., by visually separating groups from standard layers through different border styles or increased vertical spacing between parallel execution steps.

6.4 Quantitative User Study

To evaluate our final visualization design, we then conducted a quantitative study with 10 participants (3 female, 7 male, $M_{age} = 25.9$ $SD = 2.27$).

These participants were recruited in a university setting. Requirements for participation were that the participants knew what a CNN is, knew what elements CNNs consist of, and knew about feature- and spatial dimensions. Machine learning expertise of the participants varied between less than half a year and less than five years, which nicely reflects the broad audience which we expect for our visualization design. Based on internal piloting, we found that our study takes participants around an hour to complete. We compensated them with a 10 Euro Amazon gift card.

Methods. The participants were presented with different well-known machine learning architectures. Each of these architectures was visualized using three different visualization approaches, *Net2Vis* , TensorBoard, and a visualization taken from the original publication. We implemented the network architectures for each paper in Keras, which enabled us to directly export visualizations using *Net2Vis*. For TensorBoard, we had to manually screenshot and stitch the visualizations as the export functionality in TensorBoard did not work for us. For each visualization, participants had to answer eight questions by extracting information about the architecture. These questions included the following tasks: *How many convolutional layers does this architecture contain?*, *What is the maximal feature depth for the convolutional part?*, *What is the minimal spatial resolution of the convolutional part?*, *What are the input dimensions for this network?*, *What are/is the output dimension(s) of this network?*, *How many times does downsampling happen in this network?*, *How many steps are performed to increase the feature dimension?*, *Is this Architecture "Fully Convolutional"?*. Participants entered the answers to these questions into a text field and were instructed to answer *-1* if they could not extract the information from the visualization. Each participant was presented with every network architecture (6) using all three visualization techniques (3), resulting in 18 stimuli presented in a randomized order. Afterward, participants were presented with each architecture using all three visualization techniques, side by side, and were asked three comparison questions: *Which of the above visualizations contains the most useful information?*, *Which of the above visualizations was easiest to interpret?*, and *Which of the above visualizations did you find most visually appealing?*.

Analysis. To analyze the performance of the different visualization approaches, we computed the mean accuracy over all eight questions for each of the presented

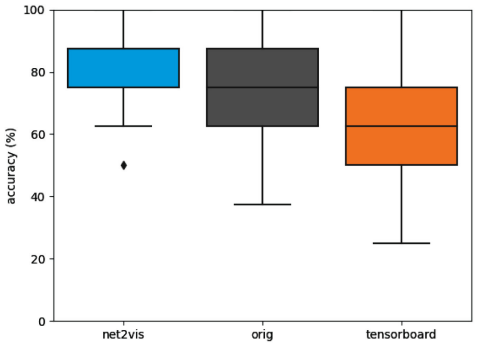

Fig. 7. This plot shows a comparison of accuracies for each condition. Accuracies are the mean of all eight questions asked for each network. For *Net2Vis*, the median is the same as the upper IQR. One can see that our approach led to the best accuracy, even though the other approaches contained questions that participants did not even need to answer. We additionally found significant differences between TensorBoard and the other two conditions, indicating that TensorBoard is the worst approach of the three for an abstract display of network architectures.

architectures for each condition (*Net2Vis* versus TensorBoard versus handcrafted), as can be seen in Fig. 7. We compared these conditions using Friedman ANOVA and found a significant difference in accuracy for the visualizations drawn using *Net2Vis* (M = 83.75 percent, SD = 12.02 percent) when compared to the handcrafted ($M =$ 75.83 percent, $SD =$ 15.75 percent), and TensorBoard ($M = 63.13$ percent, $SD = 17.29$ percent), $\chi^2(2) = 38.75$, $p <$ .001. During post-hoc analysis using Nemenyi's test, we found a significant difference between *Net2Vis* and TensorBoard ($p = .001$), as well as TensorBoard and the handcrafted version ($p = .001$). While the accuracy for *Net2Vis* was higher compared to the handcrafted version, we could not find a significant effect between these conditions ($p = .11$). When analyzing the time our participants took to complete all eight questions for each visualization, we found a significant effect between our conditions. As with accuracy, TensorBoard showed the worst performance ($M =$ 27.1s, $SD =$ 30s), followed by *Net2Vis* ($M = 16.98$s, $SD = 15$sec), and the handcrafted version ($M = 13.53$s, $SD = 8.95$s), $\chi^2(2) = 17.1$, $p <$ .001. During post-hoc analysis, we found this effect to be significant between TensorBoard versus *Net2Vis* ($p <$ .05), and TensorBoard versus handcrafted ($p <$ .001). However, when comparing *Net2Vis* and the handcrafted version, we could not find a significant effect between these conditions ($p = .23$).

For the questionnaire about which visualization technique our participants would prefer with respect to informativeness, interpretability, and design, our technique outperformed the other conditions again. For informativeness, participants preferred our approach in 86.6 percent of cases, while they favored the original version in 13.3 percent of cases. The interpretability of our design was ranked highest as well, as in 75 percent of all cases, our approach was favored, whereas the original versions were favored in 25 percent of cases. The design of our approach was favored in 70 percent of all cases, against 30 percent in favor of the original visualization. Looking at the individual networks, our approach was rated better or evenly good in all conditions except for the design of U-Net, where 60 percent favored U-Net. Remarkably, across all six conditions and all ten participants, TensorBoard did not get voted as preferable once. Used visualizations, plots, and raw study data can be found in our supplementary material, available online.

6.5 Usability Evaluation

In another evaluation with 16 participants (13 male, 1 female, 2 did not report, $M_{age} = 28.06$ $SD = 4.23$), we also evaluated our approach from the view of a visualization designer. These participants took part in our study right after a one-week full-time deep learning course, where we recruited the participants. Thus, they were familiar with the underlying concepts. Participants were given a brief introduction to *Net2Vis* before they had the chance to visualize one of their own architectures. Then, they filled a questionnaire regarding the system, including a system usability scale questionnaire (*SUS*), and a demographic questionnaire. The usability analysis through the *SUS* questionnaire resulted in a mean score of 83.44 points (SD = 6.25) indicating *excellent* usability [65].

6.6 Discussion

The main goal of *Net2Vis* was to introduce a unified design for neural network architectures as a replacement for handcrafted visualizations. While we could not find a significant effect between *Net2Vis* and the handcrafted version during our quantitative user study, we still argue that our approach outperforms the handcrafted versions in some ways. While not significant, *Net2Vis* showed overall higher accuracies, which might indicate that the effect between the conditions would become significant, given a larger number of participants. Furthermore, we saw higher accuracies in key aspects of these types of visualizations, particularly in the direct comparisons such as interpretability and design of the visualization. Besides that, for the TensorBoard visualizations and original paper figures not all information needed for answering the questions was always present during our quantitative user study. In such cases, users could answer with -1 whenever information was not available. Thus, in these cases, users could not give a wrong answer. We still counted these answers in our evaluation, essentially putting our approach at a disadvantage. Nonetheless, our techniques outperformed the others in almost all metrics. Compared to the original paper figures, which ranked better than TensorBoard, our approach has the additional advantage of a unified design and automation, which can save time and makes knowledge transfer possible.

Referring back to the nested model of evaluating visualization design [64], first, our evaluation indicates that we indeed work on a relevant problem for our target users. Moreover, the abstraction level we chose seems to be appropriate as supported by our quantitative user study. Our expert interviews clarified that it is important to keep such visualizations simple and minimalist as none of the experts complained about missing information in our visualizations. One expert even explicitly stated that it is important to *emphasize function and architecture especially over obtuse prettiness that you see in many of the tools that visualize activations or things like layer gradients*. Third, the evaluation of our

expert interviews, as well as the quantitative study, suggest that our glyph design is easily interpretable and adds valuable information as it directly visualizes the transformation of data. Fourth, the interactivity of our implementation shows that we do not have a problem with the speed of the algorithm. The results of our quantitative user study support this, with an excellent usability score for our system. While this evaluation is only a first indication of the applicability of *Net2Vis*and only adoption of the concepts in the field can prove its value, the evaluation clearly supports the need for such a tool as well as our design choices.

7 Conclusion

In this paper, we propose a visualization design for communicating neural network architectures which is based on expert feedback, state of the art visualizations, and user studies. Additionally, we provide an automated approach for visualizing CNN architectures and release a data set which contains an analysis of 751 paper figures of neural network architectures from all CVPR and ICCV conference papers since 2013. Currently, such visualizations are mostly handcrafted which consumes time in the paper writing process. Our novel visual grammar for visualizing CNNs, called *Net2Vis*, is informed by an analysis of the current practice, expert feedback, as well as widely accepted data visualization guidelines. Our proposed visual grammar incorporates visualization requirements for neural network architecture visualization, network layout, aggregation, legend generation, and a novel glyph design. *Net2Vis* represents the first visualization technique for modern and complex CNNs that is tailored towards use in publications, while the results of our quantitative user study indicate that *Net2Vis* improves both visualization generation and reading. For wide adoption, *Net2Vis* can be used as an online service at https://viscom.net2vis.uni-ulm.de, where users can obtain CNN architecture visualizations tailored towards use in publications directly from their Keras code.

Acknowledgments

This work was supported by the Carl-Zeiss Scholarship for Ph.D. students.

References

[1] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1520–1528.
[2] D. Teney and M. Hebert, "Learning to extract motion from videos in convolutional neural networks," in *Proc. Asian Conf. Comput. Vis.*, 2016, pp. 412–428.
[3] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz, "Super slomo: High quality estimation of multiple intermediate frames for video interpolation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9000–9008.
[4] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.
[5] G. Strezoski and M. Worring, "Plug-and-play interactive deep network visualization," *Proc. Vis. Anal. Deep Learn.*, 2017, pp. 100–106.
[6] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.
[7] O. Nalbach, E. Arabadzhiyska, D. Mehta, H.-P. Seidel, and T. Ritschel, "Deep shading: Convolutional neural networks for screen space shading," *Comput. Graph. Forum*, vol. 36, no. 4, pp. 65–78, 2017.
[8] P. Henzler, V. Rasche, T. Ropinski, and T. Ritschel, "Single-image tomography: 3D volumes from 2D cranial x-rays," *Comput. Graph. Forum*, vol. 37, no. 2, pp. 377–388, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13369
[9] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Conf. Operating Syst. Des. Implementation*, 2016, pp. 265–283.
[10] K. Wongsuphasawat, "Visualizing dataflow graphs of deep learning models in tensorflow," *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 1, pp. 1–12, Jan. 2018.
[11] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 675–678.
[12] D. Gschwend, "Netscope quickstart," 2017. [Online]. Available: http://dgschwend.github.io/netscope/quickstart.html
[13] F. Chollet *et al.*, "Keras," 2015. [Online]. Available: https://keras.io
[14] T. Gheorghiu, "Annvisualizer," 2018. [Online]. Available: https://github.com/Prodicode/ann-visualizer
[15] L. Roeder, "Netron," 2018. [Online]. Available: https://github.com/lutzroeder/Netron
[16] Q. Wang, J. Yuan, S. Chen, H. Su, H. Qu, and S. Liu, "Visual genealogy of deep neural networks," *IEEE Trans. Vis. Comput. Graphics*, vol. 26, no. 11, pp. 3340–3352, Nov. 2020.
[17] M. Crowe, "Neurovis," 2018. [Online]. Available: http://neurovis.mitchcrowe.com/
[18] D. Smilkov, S. Carter, D. Sculley, F. B. Viégas, and M. Wattenberg, "Direct-manipulation visualization of deep networks," *CoRR*, vol. abs/1708.03788, 2017. [Online]. Available: http://arxiv.org/abs/1708.03788
[19] A. W. Harley, "An interactive node-link visualization of convolutional neural networks," in *Proc. Int. Symp. Vis. Comput.*, 2015, pp. 867–877.
[20] M. Kahng, N. Thorat, D. H. P. Chau, F. B. Viégas, and M. Wattenberg, "GAN lab: Understanding complex deep generative models using interactive visual experimentation," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 1, pp. 310–320, Jan. 2019.
[21] M. Liu, J. Shi, K. Cao, J. Zhu, and S. Liu, "Analyzing the training processes of deep generative models," *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 1, pp. 77–87, Jan. 2018.
[22] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu, "Towards better analysis of deep convolutional neural networks," *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 1, pp. 91–100, Jan. 2017.
[23] H. Zeng, H. Haleem, X. Plantaz, N. Cao, and H. Qu, "CNNcomparator: Comparative analytics of convolutional neural networks," 2017, *arXiv : 1710.05285*.
[24] D. Bruckner, "Ml-o-scope: A diagnostic visualization system for deep machine learning pipelines," Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2014–99, 2014.
[25] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. P. Chau, "ActiVis: Visual exploration of industry-scale deep neural network models," *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 1, pp. 88–97, Jan. 2018.
[26] W. Ding, "Draw convnet," 2018. [Online]. Available: https://github.com/gwding/draw_convnet
[27] Y. Uchida, "Convnet drawer," 2019. [Online]. Available: https://github.com/yu4u/convnet-drawer
[28] A. Lenail, "Nn-svg," 2018. [Online]. Available: https://github.com/zfrenchee/NN-SVG
[29] F. M. Hohman, M. Kahng, R. Pienta, and D. H. Chau, "Visual analytics in deep learning: An interrogative survey for the next frontiers," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 8, pp. 2674–2693, Aug. 2018.
[30] J. Yuan, C. Chen, W. Yang, M. Liu, J. Xia, and S. Liu, "A survey of visual analytics techniques for machine learning," *Comput. Vis. Media*, vol. 7, no. 1, pp. 1–34, 2021.
[31] Scrapinghub, "scrapy," 2020. [Online]. Available: https://scrapy.org/
[32] PyPDF2, "Pypdf2," 2020. [Online]. Available: https://github.com/mstamy2/PyPDF2
[33] C. Clark and S. Divvala, "PDFFigures 2.0: Mining figures from research papers," in *Proc. IEEE/ACM Joint Conf. Digital Libraries*, 2016, pp. 143–152.

[34] N. Elmqvist and J.-D. Fekete, "Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines," *IEEE Trans. Vis. Comput. Graphics*, vol. 16, no. 3, pp. 439–454, May/Jun. 2010.
[35] H. Shi, J. Dong, W. Wang, Y. Qian, and X. Zhang, "SSGAN: Secure steganography based on generative adversarial networks," in *Proc. Pacific Rim Conf. Multimedia*, 2017, pp. 534–544.
[36] T. Munzner, *Visualization Analysis and Design*. Natick, MA, USA/ Boca Raton, FL, USA: AK Peters/CRC Press, 2014.
[37] M. St. John, M. B. Cowen, H. S. Smallman, and H. M. Oonk, "The use of 2D and 3D displays for shape-understanding versus relative-position tasks," *Hum. Factors*, vol. 43, no. 1, pp. 79–98, 2001.
[38] A. Cockburn and B. McKenzie, "An evaluation of cone trees," in *People and Computers XIV-Usability or Else!* Berlin, Germany: Springer, 2000, pp. 425–436.
[39] M. Brehmer, B. Lee, B. Bach, N. H. Riche, and T. Munzner, "Timelines revisited: A design space and considerations for expressive storytelling," *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 9, pp. 2151–2164, Sep. 2017.
[40] C. Felix, S. Franconeri, and E. Bertini, "Taking word clouds apart: An empirical investigation of the design space for keyword summaries," *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 1, pp. 657–666, Jan. 2018.
[41] J. Mackinlay, "Automating the design of graphical presentations of relational information," *ACM Trans. Graph.*, vol. 5, no. 2, pp. 110–141, 1986.
[42] M. Causse and C. Hurter, "The physiological user's response as a clue to assess visual variables effectiveness," in *Proc. Int. Conf. Hum. Centered Des.*, 2009, pp. 167–176.
[43] J. Heer and M. Bostock, "Crowdsourcing graphical perception: Using mechanical turk to assess visualization design," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2010, pp. 203–212.
[44] E. R. Gansner, E. Koutsofios, S. C. North, and K.-P. Vo, "A technique for drawing directed graphs," *IEEE Trans. Softw. Eng.*, vol. 19, no. 3, pp. 214–230, Mar. 1993.
[45] M. J. Proulx and H. E. Egeth, "Biased competition and visual search: The role of luminance and size contrast," *Psychol. Res.*, vol. 72, no. 1, pp. 106–113, 2008.
[46] M. J. Proulx, "Size matters: Large objects capture attention in visual search," *PloS One*, vol. 5, no. 12, 2010, Art. no. e15293.
[47] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *The Craft of Information Visualization*. Berlin, Germany: Elsevier, 2003, pp. 364–371.
[48] C. Ware, *Information Visualization: Perception for Design*. Berlin, Germany: Elsevier, 2012.
[49] R. E. Christ, "Review and analysis of color coding research for visual displays," *Hum. Factors*, vol. 17, no. 6, pp. 542–570, 1975.
[50] P. Network, "materialuicolors," 2018. [Online]. Available: https://materialuicolors.co
[51] B. Wong, "Points of view: Color blindness," *Nat. Methods*, vol. 8, 2011, Art. no. 441.
[52] W. S. Cleveland and R. McGill, "Graphical perception: Theory, experimentation, and application to the development of graphical methods," *J. Amer. Statist. Assoc.*, vol. 79, no. 387, pp. 531–554, 1984.
[53] Y. Kim and J. Heer, "Assessing effects of task and data distribution on the effectiveness of visual encodings," *Comput. Graph. Forum*, vol. 37, no. 3, pp. 157–167, 2018.
[54] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4490–4499.
[55] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Interv.*, 2015, pp. 234–241.
[56] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[57] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2818–2826.
[58] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861
[59] J. Ott, A. Atchison, P. Harnack, A. Bergh, and E. Linstead, "A deep learning approach to identifying source code in images and video," in *Proc. IEEE/ACM 15th Int. Conf. Mining Softw. Repositories*, 2018, pp. 376–386.
[60] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015. [Online]. Available: http://arxiv.org/abs/1409.1556
[61] E. Nehme, L. E. Weiss, T. Michaeli, and Y. Shechtman, "Deepstorm: Super-resolution single-molecule microscopy by deep learning," *Optica*, vol. 5, no. 4, pp. 458–464, 2018.
[62] G. Urban *et al.*, "Deep learning for drug discovery and cancer research: Automated analysis of vascularization images," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 16, no. 3, pp. 1029–1035, May/Jun. 2018.
[63] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 630–645.
[64] T. Munzner, "A nested model for visualization design and validation," *IEEE Trans. Vis. Comput. Graphics*, vol. 15, no. 6, pp. 921–928, Nov./Dec. 2009.
[65] A. Bangor, P. Kortum, and J. Miller, "Determining what individual sus scores mean: Adding an adjective rating scale," *J. Usability Stud.*, vol. 4, no. 3, pp. 114–123, 2009.

Alex Bäuerle received the master's degree in media informatics from Ulm University, in 2017. He is currently working as a research associate with the Visual Computing Group at Ulm University. His current research interests include visualization of neural networks to generate better understanding and tooling around these techniques.

Christian van Onzenoodt received the master's degree in media informatics from Ulm University, in 2017. He is currently working as a research associate with the Visual Computing Group at Ulm University. His current research interests include information visualization with a focus on the perception of visualizations.

Timo Ropinski received the PhD degree in computer science, in 2004, and the Habilitation degree, in 2009, both from the University of Münster. He is currently a professor with Ulm University, where he is heading the Visual Computing Group. Before moving to Ulm he was professor in Interactive Visualization at Linköping University, in Sweden, where he was heading the Scientific Visualization Group.